Новосибирский государственный технический университет

МЕТОДЫ КОНЕЧНОЭЛЕМЕНТНОГО АНАЛИЗА

Методические указания к лабораторным работам по курсу "Методы конечноэлементного анализа" для студентов 5-6 курсов ФПМИ, направление 010500

Составители: д.т.н., проф. *М.Г. Персова*,
д.т.н., проф. *Ю.Г. Соловейчик*,
ассист. *П.А. Домников*,
ассист. *Д.В. Вагин*

Рецензент: к.т.н., доцент *М.Г. Токарева*

Работа подготовлена на кафедре прикладной математики

ЛАБОРАТОРНАЯ РАБОТА №1

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ ДЛЯ РАБОТЫ С КОНЕЧНОЭЛЕМЕНТНЫМИ СЕТКАМИ

Цель работы

Изучить структуры данных метода конечных элементов, способы хранения информации о сетке. Научиться разрабатывать алгоритмы построения конечно-элементных сеток.

Содержание работы

Ознакомиться с фронтальным методом построения конечноэлементных сеток [1, с.474-480], методом тиражирования сечений [1, с.487-491] и методом описания области [1, с.457-460] и ее разбиения для двумерных [1, с.460-468] и трехмерных областей [1, с.480-482].

Построить конечноэлементную сетку в области, указанной в варианте задания. Для этого создать файл с описанием расчетной области и написать программу, выполняющую построение сетки. При этом для описания двумерных расчетных областей использовать формат, описанный в [1, с.480-487]. Трехмерные расчетные области задавать в файле, имеющем следующий формат.

Kx Ky // Кол-во координатных линий по x и y соответственно

x[1,1] y[1,1] ... x[1,Kx] y[1,Kx] // x y координаты точек первой координатной линии по y

x[2,1] y[2,1] ... x[2,Kx] y[2,Kx] // x y координаты точек второй координатной линии по y

... ...

x[Ky,1] y[Ky,1]...x[Ky,Kx] y[Ky,Kx] // x y координаты точек K-й координатной линии по y

```
Kz                                  // Кол-во координатных линий по z
z[1]  z[2]  ... z[Kz]               // Значения координатных линий по z

No                                  // Кол-во подобластей
m[1] nxb[1] nxe[1] nyb[1] nye[1] nzb[1] nze[1]   // Описание первой подобласти (*)
m[2] nxb[2] nxe[2] nyb[2] nye[2] nzb[2] nze[2]   // Описание второй подобласти (*)
...  ...    ...    ...    ...    ...    ...
m[No] nxb[No] nxe[No] nyb[No] nye[No] nzb[No] nze[No]    // Описание No-й подобласти (*)

// (*)
// m[i]   - Материал подобласти i
// nxb[i] - Номер x-й координатной линии начала i-й подобласти
// nxe[i] - Номер x-й координатной линии конца  i-й подобласти
// nyb[i] - Номер y-й координатной линии начала i-й подобласти
// nye[i] - Номер y-й координатной линии конца  i-й подобласти
// nzb[i] - Номер z-й координатной линии начала i-й подобласти
// nze[i] - Номер z-й координатной линии конца  i-й подобласти
```

Пример такой структуры данных приведен в [1, с.481].

Конечноэлементную сетку следует хранить как нерегулярную в общем виде с использованием двух массивов. В первом массиве хранятся координаты узлов сетки, во втором массиве – конечные элементы, в виде номеров своих узлов и номера подобласти [1, с.468-471].

Конечноэлементная сетка в области строится по файлу следующего вида, описывающего разбиение каждого интервала (по x, по y и по z) в виде количества подынтервалов и коэффициентов растяжения-сжатия (см. также [1, с.453, 462, 483]):

nx[1] cx[1] ... nx[Kx-1] cx[Kx-1] // Число подынтервалов и коэффициенты разрядки по X

ny[1] cy[1] ... ny[Ky-1] cy[Ky-1] // Число подынтервалов и коэффициенты разрядки по Y

nz[1] cz[1] ... nz[Kz-1] cz[Kz-1] // Число подынтервалов и коэффициенты разрядки по Z

Алгоритм построения конечноэлементной сетки из шестигранников по описанным выше структурам данных может быть следующим.

1. Вычислить общее количество координатных линий (включая опорные и дополнительные, определяемые разбиением каждого интервала) по x, y и z. Вычислить число узлов и элементов в сетке. Выделить память под массив, хранящий номера узлов элементов и номер подобласти, и под массив, хранящий координаты узлов сетки.
2. Написать подпрограмму, вычисляющую координаты точек разбиения одномерного интервала по заданному коэффициенту растяжения-сжатия.
3. Организовать цикл по элементарным подобластям. В этом цикле для каждой подобласти сгенерировать конечные элементы и вычислить координаты их узлов. При этом определять номера узлов элемента по его номеру, так же как при работе с регулярными сетками [1, раздел 10.2.3], организовав вложенные циклы по x, y и z. При вычислении координат узлов сетки использовать подпрограмму из предыдущего пункта. При переходе в следующую подобласть присваивать элементам соответствующий номер подобласти.

Обратим внимание на то, что при таком подходе часть узлов и ячеек может не попасть в расчетную область. Узлы регулярной сетки, не попавшие в расчётную область, обычно называют *фиктивными*. Поскольку не попавшие в расчётную область ячейки не являются конечными элементами, от них нет вкладов в глобальную матрицу. Поэтому для фиктивных узлов в конечноэлементной СЛАУ

генерируются полностью нулевые строки, и, чтобы избежать вырожденности СЛАУ, на главную диагональ таких строк (соответствующих фиктивным узлам) можно, например, поставить единицу (и тогда в фиктивных узлах после решения конечноэлементной СЛАУ получатся нулевые веса).

В вариантах заданий, где требуется построить сетку из треугольников, необходимо сначала построить сетку из четырехугольников, затем раздробить каждый четырехугольник на два треугольника. В вариантах заданий, где требуется построить сетку из прямоугольных призм с треугольным основанием, необходимо сначала построить сетку из шестигранников, затем раздробить каждый шестигранник на две призмы. Тетраэдральные сетки получать дроблением каждой призмы на три тетраэдра.

Построить изображение полученной трехмерной конечноэлементной сетки в одной из проекций путем отрисовки всех узлов и ребер сетки или любым иным способом на усмотрение студента.

Расчетные области

1. Построение четырехугольной сетки в квадранте окружности начинается с задания квадрата с обычной прямоугольной сеткой, лежащего в квадранте так, как показано на рисунке 1,а. На дуге окружности расставляется столько же узлов, сколько принадлежит двум сторонам квадрата, лежащим внутри квадранта, после чего соответствующие узлы соединяются отрезками – рисунок 1,б. На этих отрезках с некоторым коэффициентом разрядки расставляется заданное число узлов – рисунок 1,в. Последовательно соединив соответствующие узлы отрезков, получаем четырехугольную сетку – рисунок 1,г.

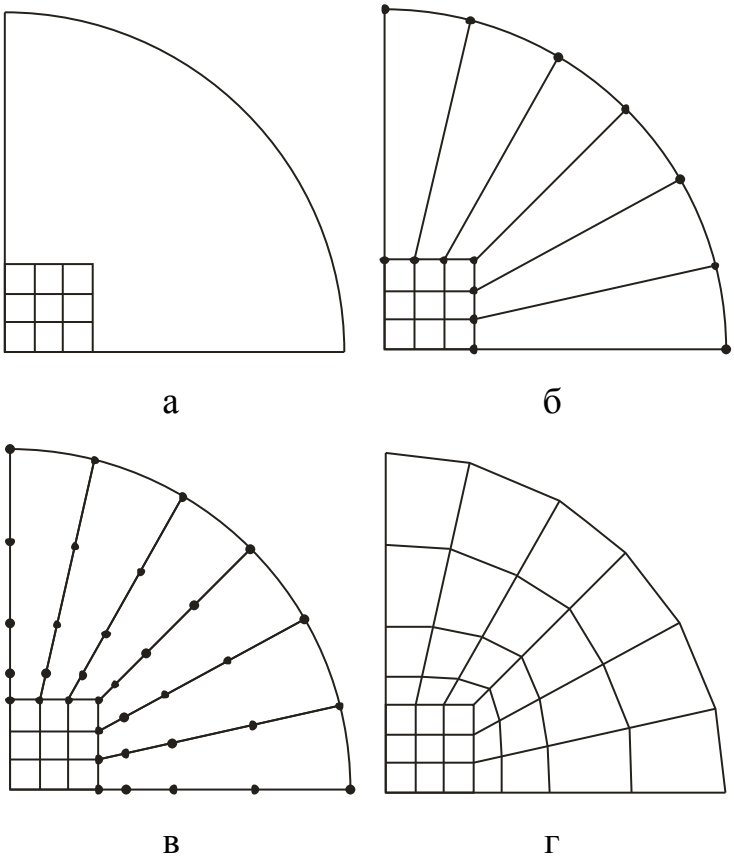

Рисунок 1. Построение четырехугольной сетки в квадранте окружности

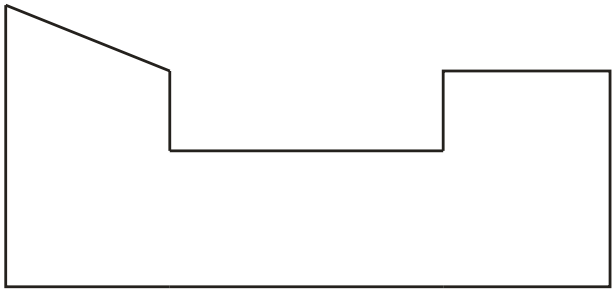

Рисунок 2.

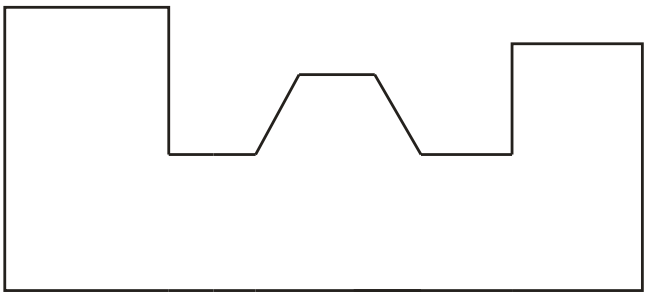

Рисунок 3.

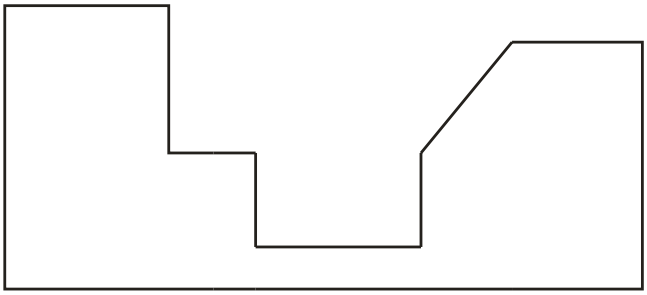

Рисунок 4.

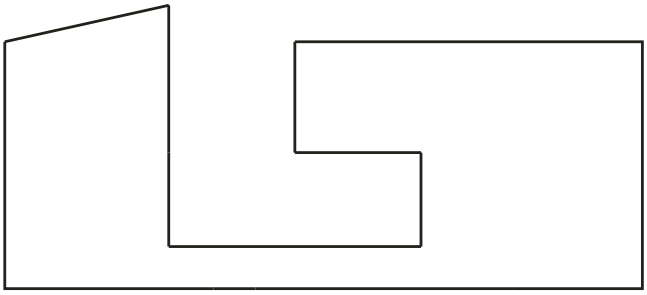

Рисунок 5.

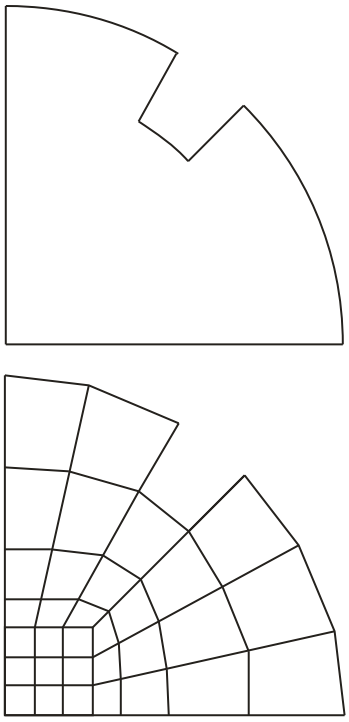

Рисунок 6.

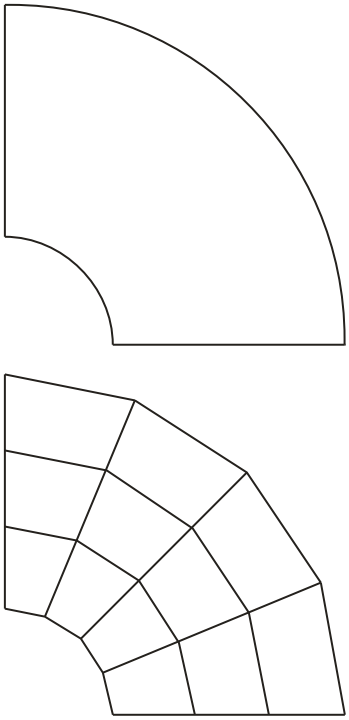

Рисунок 7.

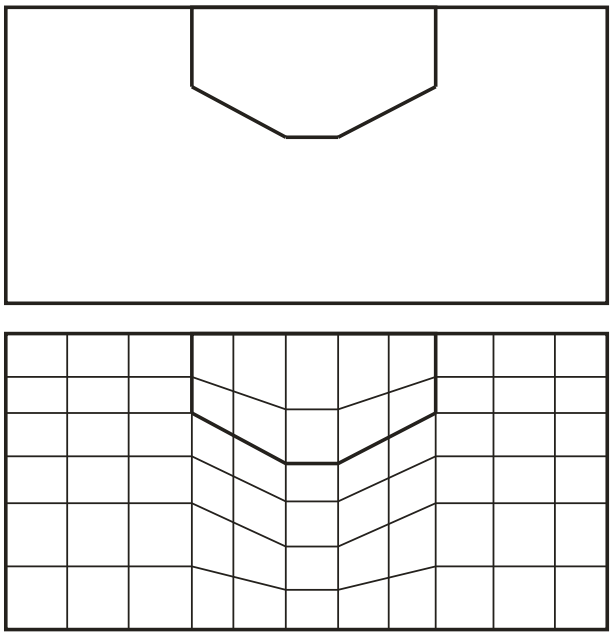

Рисунок 8.

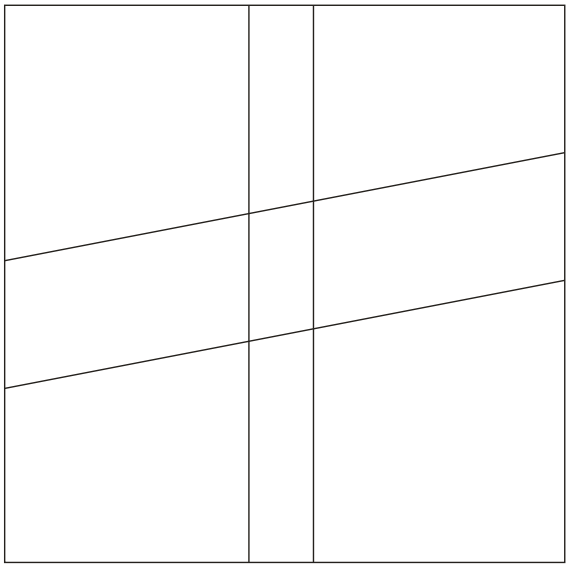

Рисунок 9.

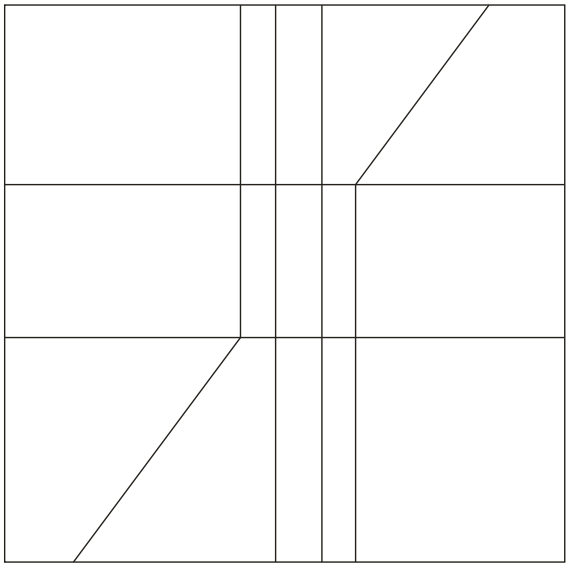

Рисунок 10.

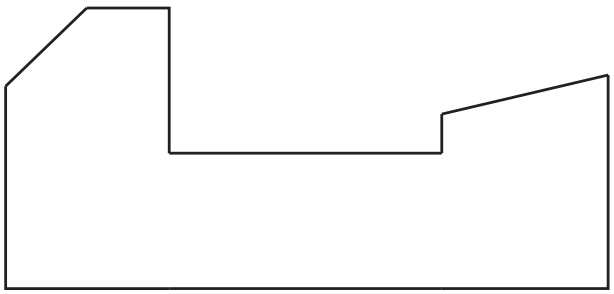

Рисунок 11.

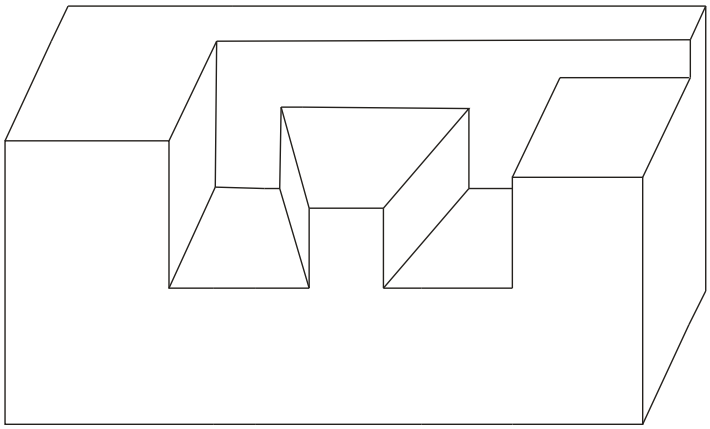

Рисунок 12.

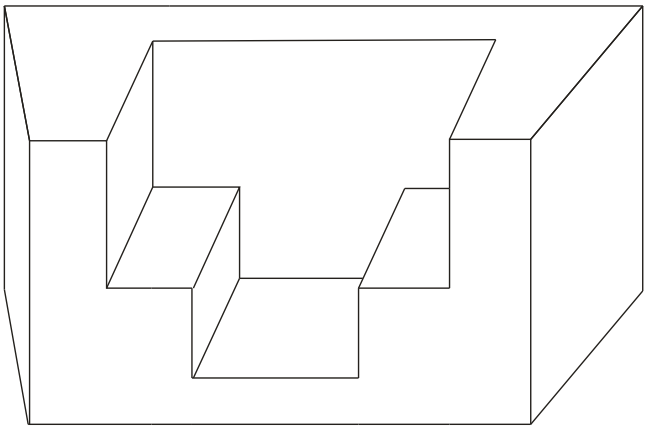

Рисунок 13.

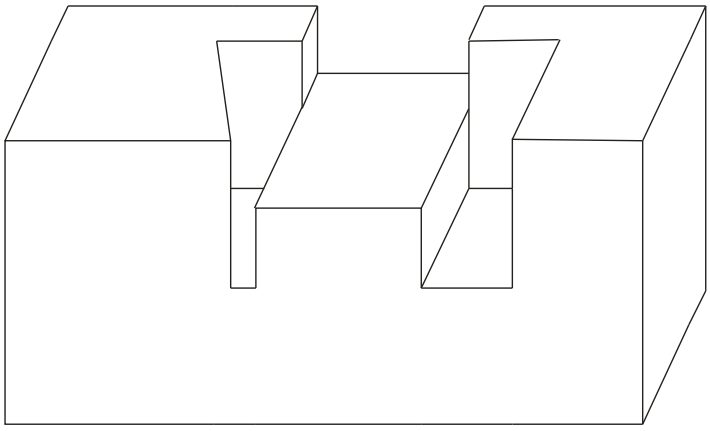

Рисунок 14.

Варианты заданий

1. Построить сетку из четырехугольников в расчетной области, изображенной на рисунке 1.
2. Построить сетку из треугольников в расчетной области, изображенной на рисунке 1.
3. Построить сетку из четырехугольников в расчетной области, изображенной на рисунке 2.
4. Построить сетку из треугольников в расчетной области, изображенной на рисунке 2.
5. Построить сетку из четырехугольников в расчетной области, изображенной на рисунке 3.
6. Построить сетку из треугольников в расчетной области, изображенной на рисунке 3.
7. Построить сетку из четырехугольников в расчетной области, изображенной на рисунке 4.
8. Построить сетку из треугольников в расчетной области, изображенной на рисунке 4.
9. Построить сетку из четырехугольников в расчетной области, изображенной на рисунке 5.
10. Построить сетку из треугольников в расчетной области, изображенной на рисунке 5.
11. Построить сетку из четырехугольников в расчетной области, изображенной на рисунке 6.
12. Построить сетку из треугольников в расчетной области, изображенной на рисунке 6.
13. Построить сетку из четырехугольников в расчетной области, изображенной на рисунке 7.
14. Построить сетку из треугольников в расчетной области, изображенной на рисунке 7.

15. Построить сетку из четырехугольников в расчетной области, изображенной на рисунке 8.
16. Построить сетку из треугольников в расчетной области, изображенной на рисунке 8.
17. Построить сетку из четырехугольников в расчетной области, изображенной на рисунке 9.
18. Построить сетку из треугольников в расчетной области, изображенной на рисунке 9.
19. Построить сетку из четырехугольников в расчетной области, изображенной на рисунке 10.
20. Построить сетку из треугольников в расчетной области, изображенной на рисунке 10.
21. Построить сетку из четырехугольников в расчетной области, изображенной на рисунке 11.
22. Построить сетку из треугольников в расчетной области, изображенной на рисунке 11.
23. Построить сетку из шестигранников в расчетной области, изображенной на рисунке 12.
24. Построить сетку из прямоугольных призм с треугольным основанием в расчетной области, изображенной на рисунке 12.
25. Построить сетку из тетраэдров в расчетной области, изображенной на рисунке 12.
26. Построить сетку из шестигранников в расчетной области, изображенной на рисунке 13.
27. Построить сетку из прямоугольных призм с треугольным основанием в расчетной области, изображенной на рисунке 13.
28. Построить сетку из тетраэдров в расчетной области, изображенной на рисунке 13.

29. Построить сетку из шестигранников в расчетной области, изображенной на рисунке 14.
30. Построить сетку из прямоугольных призм с треугольным основанием в расчетной области, изображенной на рисунке 14.
31. Построить сетку из тетраэдров в расчетной области, изображенной на рисунке 14.

ЛАБОРАТОРНАЯ РАБОТА №2

АЛГОРИТМЫ НУМЕРАЦИИ БАЗИСНЫХ ФУНКЦИЙ В МКЭ

Цель работы

Реализовать алгоритмы нумерации глобальных базисных функций в конечноэлементной сетке.

Содержание работы

Ознакомиться с алгоритмами нумерации глобальных базисных функций, построения списков ребер и граней в МКЭ [1, с.512-521].

Написать программу нумерации ребер, граней или базисных функций высокого порядка в конечноэлементной сетке согласно варианту задания. При этом использовать конечноэлементную сетку, построенную в предыдущей лабораторной работе.

Написать подпрограмму, выдающую по номеру конечного элемента номера его ребер, граней или базисных функций в соответствии с вариантом задания.

Написать подпрограмму, выдающую номер ребра по набору из двух узлов, и подпрограмму, выдающую по номеру ребра номера соответствующих узлов и конечных элементов, которым принадлежит данное ребро. Для трехмерных сеток написать также подпрограмму, выдающую номер грани по набору ее узлов, и

подпрограмму, выдающую по номеру грани номера соответствующих узлов и конечных элементов, которым принадлежит данная грань.

Варианты заданий

1. Пронумеровать ребра в конечноэлементной сетке.
2. Пронумеровать грани в конечноэлементной сетке.
3. Пронумеровать ребра и грани в конечноэлементной сетке.
4. Пронумеровать лагранжевы базисные функции второго порядка в конечноэлементной сетке.
5. Пронумеровать лагранжевы базисные функции третьего порядка в конечноэлементной сетке.
6. Пронумеровать иерархические базисные функции второго порядка в конечноэлементной сетке.
7. Пронумеровать иерархические базисные функции третьего порядка в конечноэлементной сетке.

ЛАБОРАТОРНАЯ РАБОТА №3
ПОСТРОЕНИЕ ПОРТРЕТА И СБОРКА КОНЕЧНОЭЛЕМЕНТНОЙ МАТРИЦЫ

Цель работы

Реализовать алгоритм построения портрета и сборки конечноэлементной матрицы для разреженного строчного формата хранения. Протестировать написанную программу.

Содержание работы

Ознакомиться с алгоритмом формирования портрета [1, с.487-491] и алгоритмом сборки [1, с.498-501] конечноэлементной матрицы, хранящейся в разреженном формате.

Написать подпрограмму построения портрета матрицы, возникающей при решении эллиптической краевой задачи методом конечных элементов с использованием базисных функций, указанных в варианте задания.

Реализовать алгоритм занесения локальных матриц и локальных векторов конечных элементов в глобальную матрицу и глобальный вектор конечноэлементной СЛАУ.

Проверить правильность формирования портрета и сборки конечноэлементной матрицы на тестовых задачах. При этом использовать конечноэлементную сетку, построенную в лабораторной работе №1. Для нумерации базисных функций можно использовать подпрограммы, разработанные в лабораторной работе №2.

Предусмотреть тест, в котором решением является линейная функция. На всех границах области Γ_i, не параллельных одной из координатных осей (будем считать, что это ось z), нужно задать краевое условие $u\Big|_{S_i} = cz$, где c – произвольная константа. На оставшейся части границы $S \setminus (\sum \Gamma_i)$ задаются соответствующие краевые условия второго рода. Очевидно, аналогичные тесты можно построить и для функций cx и cy.

Протестировать разработанные программы на тестах, выданных преподавателем.

Варианты заданий

1. Лагранжевы базисные функции первого порядка.
2. Лагранжевы базисные функции второго порядка.
3. Лагранжевы базисные функции третьего порядка.

4. Иерархические базисные функции второго порядка.
5. Иерархические базисные функции третьего порядка.
6. Эрмитовы базисные функции третьего порядка.

ЛАБОРАТОРНАЯ РАБОТА №4

МЕТОДЫ ПОСТРОЕНИЯ ИЗОБРАЖЕНИЙ КОНЕЧНОЭЛЕМЕНТНЫХ РЕШЕНИЙ

Цель работы

Изучить методы построения изображений конечноэлементного решения при решении различных задач. Реализовать методы построения изображения решения для элементов высоких порядков.

Практическая часть

Сгенерировать двумерную конечноэлементную сетку согласно с вариантом задания. Написать программу изображения заданной функции с помощью изолиний и заливки цветом. В качестве каркаса приложения можно использовать приведенный ниже пример программы, рисующей изолинии и изображение в виде цветовых градаций некоторой функции на поверхности, составленной из треугольных ячеек с использованием OpenGL.

```
// Структуры данных
struct Point{double x,y,z;};      // Координаты узла
struct Triangle{int nvtr[3]};     // Треугольник, определяемый номерами своих узлов
struct Map{                       // Вспомогательная структура для создания цветовой шкалы
    unsigned short red,green,blue;  // Цвет в формате RGB
    double value;                   // Значение, которому соответствует цвет
};
```

```
// Глобальные параметры
HWND hWnd;                  // Дескриптор главного окна программы
HGLRC hGLRC;                // Контекст окна для OpenGL
HDC hDC;                    // Контекст устройства

int  ktr;                           // Число треугольников
vector<double> ListOfIsolines;      // Список значений изолиний
vector<Map> Scale;                  // Цветовая шкала
vector<Point> Points;               // Список узлов
vector<Triangle> Triangles;         // Список треугольников
vector<double> q;                   // Значения изображаемой функции в узлах

int WINAPI WinMain(HINSTANCE hThisInst,HINSTANCE hPrevInst,LPSTR str,int nWinMode)
{
    MSG msg;                        // Сообщение, получаемое окном
    WNDCLASS wcl;                   // Структура, содержащая атрибуты класса окна

    InitWindow(hThisInst, nWinMode,wcl);    // Инициализация окна
    ReadData();                             // Чтение данных

    while(1){                               // Главный цикл приложения
        if(MessageLoop(msg)){               // Обработка сообщений
            // Отображение картины
            int i,j;
            // Инициализация буферов цвета
            // Три единицы – белый цвет, 0 – непрозрачность
            glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
            glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
            // Рисование поля
            for(i=0;i<ktr;i++)DrawFieldOnTriangle(i,Scale);
            // Рисование изолиний
            for(i=0;i<ktr;i++)DrawIsolineOnTriangle(i, ListOfIsolines);
            glFinish();   // Окончание рисования
            //Обмен буферов, если необходимо
            SwapBuffers(wglGetCurrentDC());
        }
        else break;
    }
    return 0;
}
```

Функция WinMain рисует изолинии и распределение некоторой функции на поверхности, составленной из треугольных ячеек, с использованием OpenGL [2]. Функция InitWindow создает и инициализирует окно приложения и настраивает его на работу с OpenGL. Далее CALLBACK-функцией WindowFunc(HWND hWnd,UINT msg,WPARAM wParam,LPARAM lParam), вызываемой системой, осуществляется инициализация OpenGL, установка проекции и вида сцены (в структуре wcl функция InitWindow присваивает соответствующему полю указатель на функцию WindowFunc). Чтение списка узлов, треугольников, значений изображаемой функции в узлах и цветовой шкалы осуществляется в функции ReadData().

В главном цикле приложения функцией MessageLoop(msg) обрабатываются сообщения, получаемые окном приложения. После этого происходит установка буферов цвета и глубины с помощью функций glClearColor и glClear. Далее следует фрагмент кода, отвечающий за рисование цветового распределения поля и изолиний.

В цикле по треугольным ячейкам вызывается функция рисования цветового распределения поля на треугольнике DrawFieldOnTriangle(i, Scale), где i – номер треугольника, Scale – цветовая шкала. Внутри этой функции для каждого узла треугольника с помощью шкалы определяется цвет, согласно значению функции в данном узле, и с помощью функций glBegin(GL_TRIANGLES), glVertex3d(x, y, z), где x, y, z – координаты точки, и glEnd() осуществляется заливка треугольника цветом. OpenGL автоматически осуществляет интерполяцию цвета между вершинами треугольника.

Далее поверх нарисованного распределения поля в цикле по треугольникам рисуются изолинии с помощью функции рисования изолинии на одном треугольнике DrawIsolineOnTriangle(i, ListOfIsolines), где i – номер треугольника, ListOfIsolines – список значений, соответствующих изолиниям. Внутри функции DrawIsolineOnTriangle(i, ListOfIsolines) выполняется проверка, проходит ли конкретная изолиния через данный треугольник, и если да, то какие его границы

она пересекает. Сама изолиния на треугольнике представляется отрезком, соединяющим точки ее пересечения со сторонами треугольника. Рисование изолиний внутри функции DrawIsolineOnTriangle осуществляется с помощью функций glBegin(GL_LINES), glVertex3d(x, y, z), где x, y, z – координаты точки, и glEnd(). Для оптимизации работы программы функции glBegin(GL_TRIANGLES), glBegin(GL_LINES) и glEnd() могут быть вызваны вне функций DrawFieldOnTriangle и DrawIsolineOnTriangle соответственно, причем всего один раз вне цикла for.

Код функций WindowFunc, DrawFieldOnTriangle, DrawIsolineOnTriangle прилагается в отдельном файле.

Варианты заданий

1. Сгенерировать сетку из плоских четырехугольников. На основании программы рисования поля на треугольной сетке написать программу для сетки из четырехугольников, разбивая каждый четырехугольник на более мелкие четырехугольники, которые затем разбить на два треугольника.
2. Сгенерировать сетку из плоских четырехугольников. На основании программы рисования поля на треугольной сетке написать программу для сетки из четырехугольников, разбивая каждый четырехугольник сначала на два треугольника, а затем на более мелкие треугольники путем проведения линий через середины ребер.
3. Сгенерировать сетку из плоских треугольников. Использовать квадратичные базисные функции. Реализовать дробление треугольников при рисовании поля и изолиний.
4. Сгенерировать сетку из плоских треугольников. Использовать кубические базисные функции. Реализовать дробление треугольников при рисовании поля и изолиний.

5. Сгенерировать сетку из плоских треугольников. Использовать иерархический базис 2-го порядка. Реализовать дробление треугольников при рисовании поля и изолиний.
6. Сгенерировать сетку из плоских треугольников. Использовать иерархический базис 3-го порядка. Реализовать дробление треугольников при рисовании поля и изолиний.

Список литературы

1. Соловейчик Ю.Г., Рояк М.Э., Персова М.Г. Метод конечных элементов для решения скалярных и векторных задач: Учеб. пособие. - Новосибирск: Изд-во НГТУ, 2007. – 896 с. («Учебники НГТУ»).
2. И. Тарасов. OpenGL [Электронный ресурс] – Режим доступа: http://opengl.org.ru/books/open_gl/index.html